access control entries in a manner analogous to the read access control discussed herein.

With convergent encryption, one encrypted version of the file is stored and replicated among the serverless distributed file system 150. Along with the encrypted version of the file is stored one or more access control entries depending upon the number of authorized users who have access. Thus, a file in the distributed file system 150 has the following structure:

$$[E_{h(F)}(F), \langle E_{K1}(h(F))\rangle, \langle E_{K2}(h(F))\rangle, \ldots, \langle E_{Km}(h(F))\rangle]$$

One advantage of convergent encryption is that the encrypted file can be evaluated by the file system to determine whether it is identical to another file without resorting to any decryption (and hence, without knowledge of any encryption keys). Unwanted duplicative files can be removed by adding the authorized user(s) access control entries to the remaining file. Another advantage is that the access control entries are very small in size, on the order of bytes as compared to possibly gigabytes for the encrypted file. As a result, the amount of overhead information that is stored in each file is reduced. This enables the property that the total space used to store the file is proportional to the space that is required to store a single encrypted file, plus a constant amount of storage for each additional authorized reader of the file.

For more information on convergent encryption, the reader is directed to co-pending U.S. Patent Application Serial No. 09/565,821, entitled "Encryption Systems and Methods for Identifying and Coalescing Identical Objects Encrypted with Different Keys", which was filed May 5, 2000, in the names of Douceur et

still pending

JH 6/2/05

al., and is commonly assigned to Microsoft Corporation. This application is hereby incorporated by reference.

For small files, the entire file is hashed and encrypted using convergent encryption, and the resulting hash value is used as the encryption key. The encrypted file can be verified without knowledge of the key or any need to decrypt the file first. For large files, the file contents are broken into smaller blocks and then convergent encryption is applied separately to each block. For example, the file F may be segmented into "n" pages $F^0$-$F^{n-1}$, where each page is a fixed size (e.g., a 4Kbyte size). Convergent encryption is then applied to the file at the block level. That is, each block $F^i$ is separately hashed using a one-way hash function (e.g., SHA, MD5, etc.) to produce a hash value $h(F^i)$. Each block $F^i$ is then encrypted using a symmetric cipher (e.g., RC4, RC2, etc.) with the hash value $h(F^i)$ as the key, or $E_{h(Fi)}(F^i)$, resulting in an array of encrypted blocks which form the contents of the file. For more information on block-by-block encryption, the reader is directed to co-pending U.S. Patent Application Serial No. 09/814,259 pending entitled "On-Disk File Format for Serverless Distributed File System", ~~Attorney Docket No. MS1-733US~~, to inventors William J. Bolosky, Gerald Cermak, Atul Adya, and John R. Douceur. This application is hereby incorporated by reference.

JH 6/8/05

File information generation module 220 can generate the file information at any of a wide variety of times. In one implementation, module 220 is designed to operate as a background process. When files are created or modified, the file names are added to a queue to be acted on by module 220. When computing device 200 is not busy (e.g., the processor has free cycles, or has been idle for a period of time), module 220 operates to generate file information for one of the files in the queue. Alternatively, module 220 may be designed to run at times of